Univerzitet u Beogradu

Fakultet organizacionih nauka

Katedra za elektonsko poslovanje

Weather App u Node.js framework-u

Seminarski rad iz Elektronskog poslovanja

Nastavnik: dr Zorica Bogdanović Student: Nevena Krstić, 10/17  
Saradnik: Luka Radovanović Nikola Jovanović, 109/17

Beograd, 2020.

SADRŽAJ

[1. UVOD 3](#_Toc30981379)

[1.1 JavaScript 3](#_Toc30981380)

[1.2 Node.js 4](#_Toc30981381)

[1.3 Zašto Node.js? 5](#_Toc30981382)

[1.4. Chat aplikacije 6](#_Toc30981383)

[2. WEATHER APP 7](#_Toc30981384)

[2.1 Projektni zadatak 7](#_Toc30981385)

[2.2 Projektovanje sistema 7](#_Toc30981386)

[2.3. Prikaz realizovanog rešenja 13](#_Toc30981387)

[3. MOGUĆA POBOLJŠANJA APLIKACIJE 17](#_Toc30981388)

[4. ZAKLJUČAK 19](#_Toc30981389)

[5. KORIŠĆENA I REFERENTNA LITERATURA 20](#_Toc30981390)

# UVOD

Za potrebe izrade projektnog rada iz predmeta Elektronsko poslovanje na trećoj godini Fakulteta organizacionih nauka, izradili smo aplikaciju za vremensku prognozu koristeći Node.js. U nastavku sledi objašnjenje rada aplikacije kao i kako smo došli do rešenja. No pre toga, želimo da vas uvedemo i u JavaScript i u sam Node.js, kako ga instalirati i gde sve nalazi praktičnu primenu.

## 1.1 JavaScript

Činjenica je da su mnogi koji ne poseduju dovoljno informacija skloni da navedu da su Java i JavaScript potpuno istovetni programski jezici. Kako bismo pokazali da nije tako, predstavićemo šta je to zapravo JavaScript i gde sve može da se koristi.

U pitanju je takozvani objektni skriptni jezik, čija je osnovna svrha korišćenja vezana za poboljšanje dinamike određene stranice. Zapravo je stvar u tome da kada se koristi HTML kod, njegova uloga je samo da oblikuje određene elemente stranice, kao što su recimo tekst, tabele, linkovi ili forma, te da ih nakon toga uredi, ali ovaj kod nema ulogu da odredi na koji način će se konkretni elementi ponašati. Upravo tada se u čitav proces uključuje JavaScript, koji ima značajnu ulogu u kreiranju ponašanja ovih elemenata.

Ovaj skriptni jezik je sačinjen od, naravno brojnih komandi, koje se moraju očitati u okviru specijalnog programa koji je nazvan Interpreter, to jest takozvani prevodilac. On služi da se te komande i očitaju, a da istovremeno ne budu prevedene u takozvanim mašinski jezik. Tačnije, tada komande bivaju iščitane direktno iz koda, pa zato i kažemo da se programski jezik JavaScript izvršava na strani korisnika. JavaScript zapravo vrši sve potrebne provere podataka već u konkretnom brauzeru i na taj način doprinosi boljem i kvalitetnijem funkcionisanju čitavog procesa. (Studio, 2019)

Smatra se upravo da je jedan od najpopularnijih skriptnih jezika koji se koristi u današnje vreme baš JavaScript, a posebno je dobro to što je on podržan od strane svih onih pregledača koji imaju najčešću primenu (Google Chrome, Mozilla Firefox, Opera, Internet Explorer i drugi). Nekoliko stvari koje je svaki prosečni korisnik interneta video su:

* padajući meni
* neobični pokazivač miša
* iskačući prozor
* sat

i slično

Međutim postoje ozbiljnije primjene JavaScript-a kao što su:

* Detekcija pregledača
* Kontrola prozora
* Web kolačići (takozvani kukiji)
* Provera sadržaja

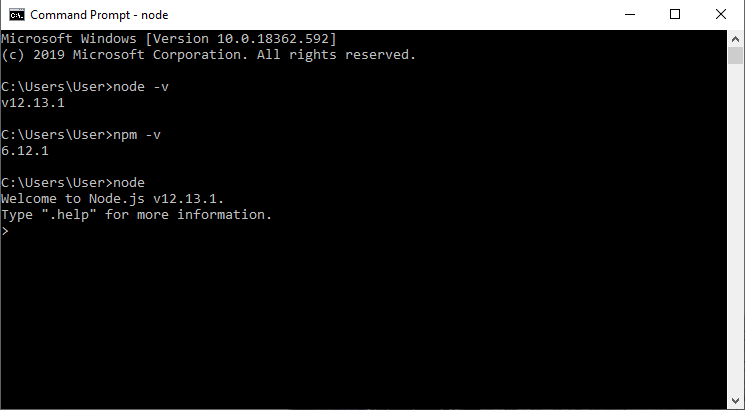
## 1.2 Node.js

Do sada smo koristili JavaScript jezik u jednom okruženju: pregledaču. Ovo poglavlje će ukratko predstaviti Node.js, program koji nam omogućava da primenimo naše JavaScript veštine van pretraživača. Pomoću njega možemo da napravimo bilo šta, od malih alata komandne linije do HTTP servera koji pokreću dinamičke veb lokacije. Mi smo u našem slučaju napravili aplikaciju za vremensku prognozu, odnosno takozvani WeatherApp služeći se upravo Node.js-om. (znanje.org, 2018)

Kako bismo uspeli da odradimo našu aplikaciju, prvobitno smo sa ovog linka: <https://nodejs.org/en/> instalirali Node.js.

Kada je Node.js instaliran na naš računar, pruža nam program koji se zove node, a koji se koristi za pokretanje JavaScript datoteka.

Posle instalacije, kako bi proverili da li je node.js ispravno instaliran, odemo u command prompt i ukucamo node -v, ili node –version. Ako je instalacija uspesno izvrsena na ekranu bi trebali da vidimo verziju instaliranog node-a. Posle toga cemo proveriti verziju NPM-a, komandom npm -v.



Slika 1: Verzija node.js-a

## 1.3 Zašto Node.js?

Kao prvo omogućava korišćenje jedinstvenog jezika između front-end i back-end-a. To znači da kompletnu aplikaciju možete realizovati pomoću jednog programskog jezika.

Drugo, Brzina – Brz je. Zajedno sa Guglovim V8 JavaScript engine-om, ima neverovatnu brzinu. Zapravo, nekoliko puta je brži od ostalih skriptnih jezika poput Ruby-a i Python-a.

Treće, njegova neblokirajuća arhitektura je idealna za pravljenje real-time web aplikacija. JavaScript je izabran kao jezik zbog svog malog core API-ja i nonblocking callbacks-a. To znači da je moguće izgraditi kompletan Node.js ekosistem oko neblokirajućih paketa, kojih trenutno ima više od deset hiljada. Krajnji rezultat je platforma i ekosistem koji se arhitekturalno savršeno uklapa sa zahtevima izrade modernih real-time web aplikacija. (Petlja, 2018)

## 1.4. Chat aplikacije

Chat je tipičan primer real-time, multi-user aplikacije i najbolji primer gde Node.js dolazi do izražaja. Kako to otprilike izgleda: u nekom najjednostavnijem primeru imamo chatroom na našem web sajtu gde korisnici mogu da razmenjuju poruke.

Sa serverske strane imamo Express.js aplikaciju koja implementira dve stvari:

* GET “/” request handler koji opslužuje veb stranicu sa porukom i dugmetom za slanje i inicijalizovanje nove poruke.
* Websocket server koji sluša novu poruku koju šalje websocket client.

Sa klijentske strane se nalazi HTML stranica sa nekoliko podešenih handlera.

# WEATHER APP

## 2.1 Projektni zadatak

Tema našeg projektnog zadatka jeste izrada Weather App (aplikacije za prikaz vremenske prognoze) u Node.js framework-u. Ovo smo realizovali tako što smo napravili aplikaciju koja izbacuje trenutnu temperaturu određenog grada koji smo uneli u polje za unos.

Koristili smo OpenWeatherMap servis koji nam šalje informacije o vremenskoj prognozi za zadati grad, i trenutna temperatura se ispisuje na ekranu. Za izradu web aplikacije smo koristili Express.js, koji se jednostavno instalira i koristi. Za izradu web strane koristili smo EJS (Embedded JavaScript), u kojem se sve promenljive i kod konvertuju u HTML dokument prilikom prikazivanja veb stranice.

Aplikacija je klijent-server aplikacija, gde na zahtev klijenta za određenim gradom, server šalje informacije o vremenskoj prognozi i ispisuje ih na ekranu. Zbog toga se glavna logika aplikacije odvija na serveru, tj u server.js fajlu. (Lambert M. Surhone, 2010)

## 2.2 Projektovanje sistema

Za početak, našu aplikaciju, tačnije kod aplikacije možete naći na git-u na sledećem linku (<https://github.com/nekicaa/weatherapp>). U repozitorijumu postoje sledeći fajlovi i folderi:

* node modules (folder, napravljen po default-u instalacijom potrebnih dodataka)
* public/css (folder u kom se nalazi css fajl za određivanje izgleda stranice)
* views (folder u kom se nalazi index.ejs fajl za određivanje elemenata stranice)
* package-lock.json
* package.json (JSON fajlovi napravljeni po default-u instalacijom dodataka)
* server.js (ključni fajl aplikacije, serverska strana)

Analizu koda ćemo početi od index.ejs fajla koji je upotrebljen za kreiranje izgleda veb stranice. Naš ejs fajl izgleda ovako:

<!DOCTYPE html>

<html>

  <head>

    <meta charset="utf-8">

    <title>Weather App</title>

    <link rel="stylesheet" type="text/css" href="/css/style.css">

    <link href='https://fonts.googleapis.com/css?family=Open+Sans:300' rel='stylesheet' type='text/css'>

  </head>

  <body>

    <div class="container">

      <fieldset>

        <form action="/" method="post">

          <input name="city" type="text" class="ghost-input" placeholder="Unesite ime grada" required>

          <input type="submit" class="ghost-button" value="Pronađi trenutnu temperaturu">

        </form>

        <% if(weather !== null){ %>

          <p><%= weather %></p>

        <% } %>

        <% if(error !== null){ %>

          <p><%= error %></p>

        <% } %>

      </fieldset>

    </div>

  </body>

</html>

Naizgled izgleda isto kao i HTML fajl, koji sadrži “container” sa dva input polja, jedno polje za unos teksta i drugo polje koje je “button” tipa, tj. dugme. Ovaj index.ejs fajl je povezan sa style.css fajlom u folderu css, koji formatira stranu da izgleda baš tako kako izgleda.

Ono u čemu se razlikuje ovaj tip fajla od običnog HTML-a jeste baš ovaj deo u <% … %> zagradama, koji mu daje mogućnost da se ponaša kao JavaScript fajl. Konkretno u našem primeru se proverava da li je vrednost promenljive *weather* u JS fajlu jednaka null. Ako nije, tj. ako smo dobili neku povratnu vrednost za vreme, onda se ispisuje odgovarajuća poruka koja je smeštena u toj promenljivoj. Isto tako proveravamo i promenljivu *error*, koja u sebi čuva poruku koja se izbacuje ukoliko dođe do greške. Ako dođe do greške onda ta poruka, tačnije promenljiva nije jednaka null, pa se u tom slučaju ispisuje data vrednost, tj. poruka o grešci.

Naša stranica je jednostavna sa jednostavnim izgledom, određenim sledećim css fajlom :

body {

    width: 800px;

    margin: 0 auto;

    font-family: 'Open Sans', sans-serif;

    background-image: url("bkground.jpg");

    background-position: center;

    background-repeat: no-repeat;

    background-attachment: fixed;

    background-size: cover;

  }

  .container {

    width: 600px;

    margin: 0 auto;

  }

  fieldset {

    display: block;

    -webkit-margin-start: 0px;

    -webkit-margin-end: 0px;

    -webkit-padding-before: 0em;

    -webkit-padding-start: 0em;

    -webkit-padding-end: 0em;

    -webkit-padding-after: 0em;

    border: 0px;

    border-image-source: initial;

    border-image-slice: initial;

    border-image-width: initial;

    border-image-outset: initial;

    min-width: -webkit-min-content;

    padding: 30px;

  }

  .ghost-input, p {

    display: block;

    font-weight:300;

    width: 100%;

    font-size: 25px;

    border:0px;

    outline: none;

    width: 100%;

    -webkit-box-sizing: border-box;

    -moz-box-sizing: border-box;

    box-sizing: border-box;

    color: #4b545f;

    background: #fff;

    font-family: Open Sans,Verdana;

    padding: 10px 15px;

    margin: 30px 0px;

    -webkit-transition: all 0.1s ease-in-out;

    -moz-transition: all 0.1s ease-in-out;

    -ms-transition: all 0.1s ease-in-out;

    -o-transition: all 0.1s ease-in-out;

    transition: all 0.1s ease-in-out;

  }

  .ghost-input:focus {

    border-bottom:1px solid #ddd;

  }

  .ghost-button {

    background-color: white;

    opacity: 0.5;

    border:2px solid #ddd;

    padding:10px 30px;

    width: 100%;

    min-width: 350px;

    -webkit-transition: all 0.1s ease-in-out;

    -moz-transition: all 0.1s ease-in-out;

    -ms-transition: all 0.1s ease-in-out;

    -o-transition: all 0.1s ease-in-out;

    transition: all 0.1s ease-in-out;

    color: black;

  }

  .ghost-button:hover {

    border:2px solid #515151;

    opacity: 0.8;

    font-size: large;

  }

  p {

    background-color: transparent;

    color: rgb(253, 170, 36);

  }

U ovom fajlu je definisana pozadina, boja teksta i ostalih elemenata, kao i njihove pozicije i prikaz na ekranu.

Kao što smo već napomenuli, cela logika se zapravo odvija na serverskoj strani, tačnije u server.js fajlu koji izgleda ovako:

const express = require('express');

const bodyParser = require('body-parser');

const request = require('request');

const apiKey = 'dbb281de6497fc87a869a028b4aa038c';

const app = express()

app.use(express.static('public'));

app.use(bodyParser.urlencoded({ extended: true }));

app.set('view engine', 'ejs')

app.get('/', function (req, res) {

  res.render('index', {weather: null, error: null});

})

app.post('/', function (req, res) {

let city = req.body.city;

let url = `http://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&appid=${apiKey}`

request(url, function (err, response, body) {

  if(err){

    res.render('index', {weather: null, error: 'Greška, molimo Vas pokušajte ponovo'});

  } else {

    let weather = JSON.parse(body)

    if(weather.main == undefined){

      res.render('index', {weather: null, error: 'Greška, molimo Vas pokušajte ponovo'});

    }else{

      let weatherText = `Sada je ${weather.main.temp} stepeni u ${weather.name}!`;

        res.render('index', {weather: weatherText, error: null});

    }

  }

});

})

app.listen(3000, function () {

  console.log('Weather app listening on port 3000!')

})

Ovaj kod ćemo analizirati redom, linija po linija, od početka do kraja. Za početak napravimo promenljive koje će se kasnije koristiti i to su: *express*, *bodyParser*, *request*, *apiKey* i *app*. Prve tri se odnose na pozivanje paketa koje smo prethodno instalirali preko terminala i komandi *npm install –save \**ime komande*\**. Instanca app poziva Express JS. *ApiKey* se odnosi na API ključ, kako ne bismo istu vrednost koju koristimo kopirali više puta u kodu.

Vrednost *apikey*-a, tj API ključ smo dobili preko stranice OpenWeatherMap (<https://openweathermap.org>) gde se jednostavno nakon registracije napravi i dobije jedinstven API ključ koji je spreman za upotrebu u projektu. Preko njega se uzimaju određene vrednosti traženih prognoza sa OpenWeatherMap platforme sa koje povlačimo potrebne informacije o vremenu.

Da bi Express dozvolio pristup css fajlu, potrebna je naredna linija koda: app.use(express.static('public'));

BodyParser koristi narednu liniju koda kako bi mogao da zahteva da povlači vrednosti unete u napravljenu formu u ejs fajlu: app.use(bodyParser.urlencoded({ extended: true }));

Kako bi server učitao našu stranicu onako kako smo je definisali u ejs i css fajlovima koristi se sledeća linija koda: app.set('view engine', 'ejs')

Deo koda app.get( … )  se koristi za slanje određene poruke klijentu, u ovom slučaju određene strane. Koristimo funkciju render u ovom slučaju kako bi se ejs fajl, tj stranica kod klijenta prevela u odgovarajuću HTML stranu i tako učitala.

Deo koji se nalazi u app.post( … ) delu se odnosi na serverski odgovor klijentu. Najpre u njemu definišemo promenljive koje se odnose na grad za koji tražimo prognozu tj. promenljiva *city*, i promenljiva *url* koja se odnosi na veb adresu sa koje zahtevamo da nam podaci o vremenskoj prognozi budu poslati. Pored prosleđivanja unetog grada datoj adresi, prosleđuje se i API ključ koji se koristi i u našem slučaju smo definisali da temperaturu želimo da dobijemo izraženu u stepenima Celzijusove skale (‘units=metric’).

U delu request(url, function (err, response, body) { ... } smo formirali zahtev da se sa prethodno prosleđene url adrese izvuku potrebni podaci. Za početak ukoliko se javi neka greška treba ispisati datu poruku o njoj. Ako nema greške, onda se porverava da li je unet grad validan, tj proverava se da li postoji taj grad. U slučaju nevalidnog unosa imena grada izbacuje se poruka o grešci. Ukoliko je unet grad ispravno napisan, onda se izbacuje odgovarajuća poruka o prognozi, tj. temperaturi u tom gradu. *Napomena: obzirom da u odgovoru koji server šalje klijenu koristi podatke iz baze, ime grada koji se prosleđuje se prikazuje na engleskom jeziku jer u toj formi postoji u bazi.*

Na kraju, linija koda koja se odnosi na osluškivanje servera za zahtevima klijenta je u delu:

app.listen(3000, function () {

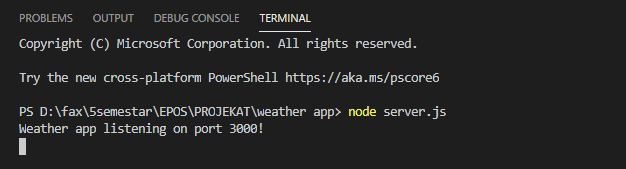
  console.log('Weather app listening on port 3000!')

})

Ovde se na portu 3000 čekaju korisnički zahtevi.

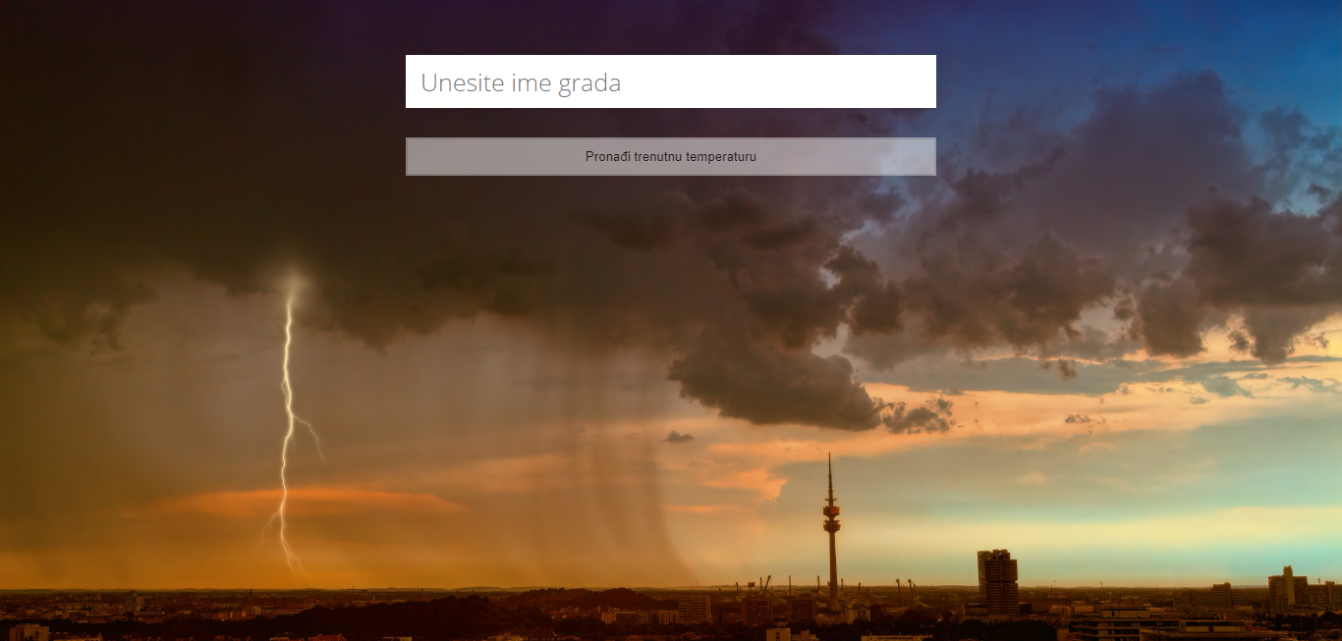
## 2.3. Prikaz realizovanog rešenja

Za pozivanje aplikacije, kao što smo u prethodnom delu naveli, potrebno je da inicijalizujemo osluškivanje servera na portu 3000. To se radi tako što u terminalu pozovemo *node server.js* i ukoliko nam on da odgovarajuću poruku da je server spreman, na adredi *localhost:3000* možemo da učitamo našu stranicu.



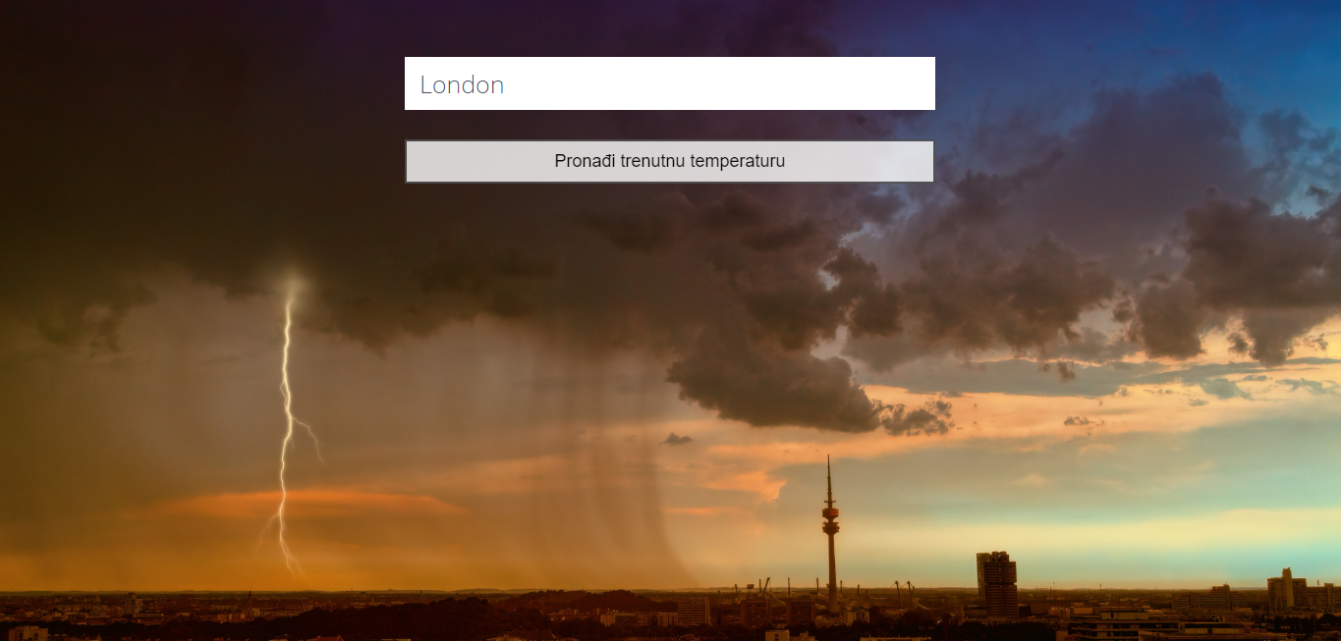
Slika 2: Inicijalizacija servera za osluškivanje

Dobili smo odgovarajuću poruku *Weather app listening on port 3000!* pa sada na adresi *localhost:3000* dobijamo sledeću stranicu:

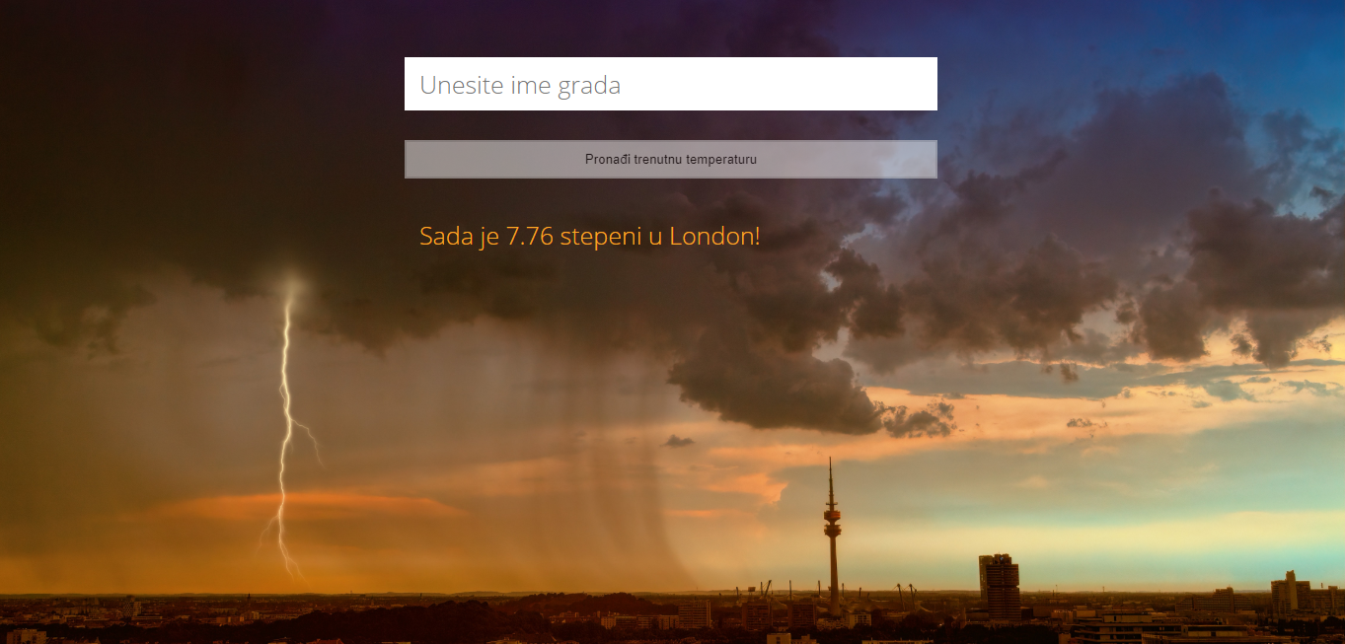


Slika 3: Početna strana

Ukoliko u polje unosa imena grada unesemo npr. London i kliknemo na dugme ispod (slika 4), dobićemo sledeći odgovor (slika 5).



Slika 4: Unos imena grada (primer London)

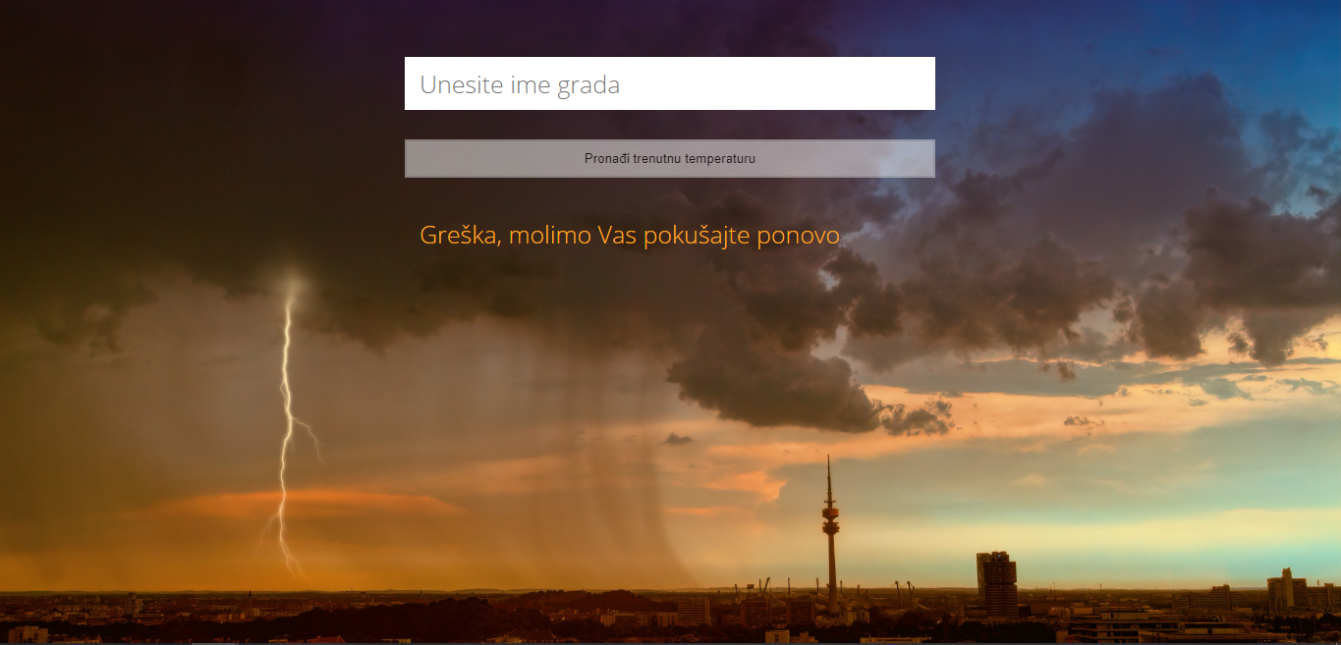


Slika 5: Temperatura u unetom gradu

Ukoliko umesto pravog imena grada, unesemo pogrešno ime ili neki niz slova ili brojeva koji nije grad, kao na sledećoj slici (slika 6), izbaciće se poruka o grešci na ekranu (slika 7).



Slika 6: Pogrešan unos



Slika 7: Poruka o grešci

# MOGUĆA POBOLJŠANJA APLIKACIJE

Aplikacija koju smo napravili je jednostavna, ali praktična. Na lak način za određen korisnički unos daje informacije o trenutnoj temperaturi u nekom gradu. Ali takođe i postoji dosta prostora za poboljšanje iste, a sada ćemo navesti par primera na koja smo naišli i razmislili o njima u toku rada.

Za početak, aplikacija bi mogla da dostavlja korisniku i neke druge informacije koje se tiču vremenske prognoze, a da to nije samo trenutna temperatura. Prilikom zahteva za pristup i prosleđivanje informacija o prognozi, mi smo u kodu striktni definisali da želimo da nam se prosledi i ispiše isključivo trenutna temperatura. Međutim ukoliko ne definišemo konkretno vrednost koju uzimamo, sa OpenWeatherMap-a nam se prosleđuju sve vrednosti koje postoje za prognozu u tom gradu, kao što su informacije o tome kakvo je vreme (sunčano, oblačno, kišovito …), informacije o vetru (kolikom brzinom i iz kog smera se kreće), informacije o vlažnosti vazduha, pritisku, minimalnoj i maksimalnoj dnevnoj temperaturi, vidljivosti itd. Za kreiranje kompleksnije aplikacije, sve ove informacije bi bile od značaja i mogle bi da se iskoriste kako bismo klijentu izneli detaljniji izveštaj o vremenu.

Možda bi neki korisnici naše aplikacije bili česti klijenti, pa bi mogla da se napravi forma za registraciju gde će registrovanim korisnicima biti uvek prikazani podaci o izabranim gradovima.

Takođe, uz registraciju korisnika, bilo bi korisno da se uvede deo za komentare, koje će svi klijenti koji traže temperaturu određenog grada mogli da vide i dobiju uvid u lične utiske ostalih korisnika o vremenu. Komentari bi bili tipa “Iako ima sunca, u hladu je dosta hladnije i duva vetar.”, “Sneg se jako brzo topi, tako da je po ulicama sve mokro. Preporuka je da nosite nepromočivu obuću.”, “Danas je sunce jače nego inače iako po temperaturi ne deluje tako. Lako se može dobiti sunčanica, zato bolje nosite kapu” i sl. To bi doprinelo realnom utisku u gradovima ili delovima grada, jer npr. dešavalo se da na prognozi dobijemo jednu informaciju o vremenu u Beogradu, dok su zapravo uslovi na Dorćolu i Banovom brdu drastično različiti.

Još jedna stvar koja bi verovatno olakšala korišćenje aplikacije je postavljanje mape na veb stranici i omogućavanje korisnicima da izborom neke lokacije sa mape dobiju informacije o vremenskoj prognozi korišćenjem unetih koordinata (tačnije odabranih koordinata).

Naravno da ima još dosta primera gde bi sve mogla da se usavrši aplikacija, ali mi smo predstavili jednostavan primer koji se najčešće koristi i jednistavan je za upotrebu.

# ZAKLJUČAK

Ovom veb aplikacijom pokazali smo da Node.js znatno olakšava posao i da može biti dosta koristan pri izradi ovakvih ili sličnih aplikacija. Nije mnogo težak za učenje i vrlo brzo je moguće savladati neke osnovne funkcionalnosti kako bi se napravila aplikacija ovog tipa. Uz još malo dodatnog istraživanja, produbljivanja u Node.js moguće je za vrlo kratko vreme steći dovoljno znanja kako bi se napravila i neka malo kompleksnija aplikacija, kao što smo na primer u prethodnom odeljku naveli, aplikacija koja bi nam pored trenutne temperature pokazivala i komentare korisnika, brzinu vetra, vlažnost vazduha i slično.

# 5. KORIŠĆENA I REFERENTNA LITERATURA

*Petlja*. (2018). Retrieved from Petlja: https://petlja.org/biblioteka/r/lekcije/prirucnik-js/js-u-node-js

*znanje.org*. (2018). Retrieved from znanje.org: https://www.znanje.org/knjige/computer/JavaScript/2010/index.htm

Lambert M. Surhone, M. T. (2010). *Node.js.* Beau Bassin, Mauritius: Betascript Publishing.

*openweathermap*. (n.d.). Retrieved from openweathermap: openweathermap.org

Studio, O. W. (2019). *Šta je JavaScript*. Retrieved January 2020, from oxfordwebstudio: https://www.oxfordwebstudio.com/da-li-znate/sta-je-javascript.html